System and Method For Collaborative Data Sharing

Inventors:

Vani Kola and Kerman D. Deboo

BACKGROUND OF THE INVENTION

1. Field of the Invention.

The present invention relates generally to the collaborative development and use of data in computer systems. In particular, the present invention relates to a system and method for creating and modifying collaboratively shared data. Still more particularly, the present invention relates to creating and modifying collaboratively shared data in an automated system for conducting business transactions for the purchase and sale of non-production goods.

2. Description of the Background Art.

Over the last decade, computer systems have undergone a gradual evolution from the single mainframe to distributed networks of personal computers and workstations. Because computer networks are extremely efficient at maintaining and communicating information and performing activities between distributed sites, modern businesses should be the obvious beneficiaries of this revolution in technology. For example, databases used by different departments within a business could be created, maintained and modified automatically over a shared computer network.

One example where collaborative data sharing is essential is the purchase of non-production goods and services, often called MRO purchasing. At present, most business to business commerce is not automated and generally paper based. For example, a corporation's purchase of an item for a particular task is usually a manual process and is often very labor intensive. In the typical purchasing process 1) data needs to be gathered about the item to be purchased, 2) potential vendors need to be identified, 3) prices need to be compared, 4) a paper based requisition needs to be filled out and submitted for approval, 5) the purchase requisition needs to be routed to a purchasing department for further scrutiny and approval, 6) the purchase requisition needs to be transferred or converted into a purchase order, 7) the purchase order needs to be approved and then submitted to the vendor

for fulfillment, and 8) the order needs to be tracked until the goods are delivered. This process is subject to much error due to the numerous manual steps in the process, the data transcription requirements, and the many different individuals that need to participate in routing the documents through the process.

Thus, collaborative data sharing for use in automation of procurement has long been an illusive goal for many businesses. Some electronic resource planning (ERP) and electronic data interchange (EDI) systems have addressed part of the problem, but those solutions generally address specialist purchasing activities for a small user population in a business. Automation of procurement for the enterprise by allowing users to purchase those items they need, when they need them, and to significantly reduce the cost of that purchasing process has not been successfully addressed in the prior art. Moreover, numerous shortcomings in the state of the art prevent businesses from fully exploiting conventional network technology for collaborative data sharing in such an application.

One particular problem specific to the collaborative sharing of data is the creation, modification, organization and dissemination of information maintained in shared databases in a consistent and orderly manner. The prior art has attempted to reduce such problems through the use of database replication programs. Database replication is a method for modifying, adding to or deleting data in a database. With conventional database replication, a user may access and download data from a master database. Once the data is downloaded, the user may modify, delete or add to the downloaded data. The modified data is then compared against and merged with the master database.

However, conventional database replication programs suffer from a number of shortcomings. First, many different changes to data in a database may be entered by more than one user often resulting in multiple, concurrent, and conflicting changes. Prior art database replication systems do not always ensure that shared data modified by different users remain current, correct and consistent. Moreover, current database replication programs do not provide for verification and approval of changes before distributing the changes to the target systems. Another problem is that conventional database replication methods are usually implemented immediately and do not provide for modifications at pre-scheduled times. Additionally, it is often desirable to modify data a single time and have the

changes implemented in various separate target systems rather than having to modify the data for each particular system. Finally, current replication methods only address data in the database and cannot deal with data objects outside the database structure, e.g. images left in a general file system directory.

Therefore, there is a continuing need for a system and method for collaborative data sharing in a multi-tier computer network. More particularly, there is a need for an automated system that utilizes such system and methods for collaborative data sharing in a multi-tier computer network for automated procurement.

SUMMARY OF THE INVENTION

The present invention overcomes the deficiencies and limitations of the prior art with a system for collaborative data sharing in a multi-tier computer network. The present invention is a system for performing a release process for creating, maintaining and modifying shared data in a collaborative environment and more specifically, for modifying data in a staging system to affect changes in an on-line system. The system for performing a release process in accordance with the present invention preferably comprises a release manager module, a staging system and an on-line system. The on-line system allows users to view and retrieve information. The present invention operates such that there may be a plurality of such on-line systems. The staging system provides access to users desiring to create or modify data in an on-line system. These users can modify the data in the database and specify when such changes should be made to any one of a number of on-line systems that are in existence. The release manager module controls the updating of the on-line system from the staging system and can perform incremental updating to only portions of the on-line system or full updating where all modified portions of the on-line system are made current.

The present invention also comprises a method for performing a release process for modifying data in a staging system to affect changes in an on-line system. The preferred method for performing a release process comprises the steps of: updating a record in the staging system; packaging the record into an update package; transferring the update package from the staging system to the on-line system; and updating the on-line system

using the update package. The present invention also includes methods for maintaining data consistency during the step of updating a record in the staging system and on-line system.

One embodiment of the present invention is a procurement system for automating the procurement and purchase of items. A procurement system embodying the principles of the present invention preferably includes a release manager module for performing a release process and preferably operates on a computer network that includes one or more computers and a legacy or mainframe computer coupled by a computer network and/or the Internet/Intranet. The procurement system is preferably a web-centric application that comprises a thin client, a Web Server, and an application server and a database server. The procurement system also includes at least one staging system and at least one on-line system, each of which stores information used by the procurement system. The on-line system allows users to view and retrieve current information, and place and process orders. The staging system is accessible to authors of information, such as authors of data in a catalog, or authors of processes for handling orders. The release manager module controls the updating of the on-line system from the staging system. A procurement system which includes a release manager module allows users to modify data in the staging system and specify when such changes should be made to any one of a number of on-line systems that are in existence.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a graphical representation of transaction processes involved in a procurement system operating according to the present invention;

Figure 2 is a block diagram of a computer system constructed according to the present invention including a procurement system;

Figure 3 is a block diagram of a preferred embodiment of the memory including the components of the procurement system according to the present invention;

Figure 4 is a block diagram of a preferred embodiment of the staging system and on-line system constructed according to the present invention.

Figure 5 is detailed block diagram of the preferred embodiment of the purchase manager constructed according to the present invention;

Figure 6 is a detailed block diagram of the preferred embodiment of the functional components of the procurement system constructed according to the present invention;

Figure 7 is a block diagram graphically showing the software architecture for the procurement system of the present invention;

Figure 8 is a flowchart of a preferred method for performing a release process in accordance with the present invention;

Figure 9 is a flowchart of a preferred method for updating the staging system in accordance with the present invention;

Figure 10 is a flowchart of a preferred method for maintaining data consistency during the staging system of the release process in accordance with the present invention;

Figure 11 is a flowchart of a preferred method for packaging data scheduled to be released in accordance with the present invention;

Figure 12 is a flowchart of a preferred method for transferring packaged data to a target on-line system in accordance with the present invention; and

Figure 13 is a flowchart of a preferred method for an on-line updating process in accordance with the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

While the present invention will now be described in terms of a procurement system, those skilled in the art will realize that the underlying structure and processes of the present invention apply to a variety of other business applications such as human resources, real estate, logistics, asset management, expense reporting, time card management, cross - enterprise management, and others, and the procurement system is provided only by way of example.

Referring to Figure 1, a block diagram of a system 100 embodying the principles of the present invention is shown. A plurality of sites, 102, 104, 106, 108, 110, 112, and 114, are coupled to a network 116. In a preferred embodiment, system 100 is a procurement system and sites 102, 104, 106, 108, 110, 112, and 114 are business enterprises which act as customers, manufacturers, suppliers, and/or distributors. For example, site 102, acting as a customer site, accesses site 104, a multi-vendor catalog site, and searches or browses the catalog for a particular item. When site 102 has selected an item for purchase, site 102 accesses site 106 and fills out a purchase requisition or purchase order. Site 106 submits the purchase requisition or purchase order to site 108, a vendor purchasing department, for approval. The purchase order is then submitted to site 112, a manufacturer or supplier, in

order to fill the purchase order. Site 114 allows the customer to keep track of the order on-line. Sites 102, 104, 106, 108, 110, 112, and 114 communicate with each other over network 116. Each of sites 102, 104, 106, 108, 110, 112, and 114 is a computer or computer system as will be described below in more detail with reference to Figure 2. Network 116 may be any type of computer network system such as an intranet, the Internet, or an Extranet, and is preferably the Internet.

System 100 is preferably a Web-centric, multi-tier application that allows connecting sites 102, 104, 106, 108, 110, 112, and 114 to work collaboratively to complete a business transaction. One advantage of the present system is its ability to eliminate errors caused by prior art manual procedures. Additionally, system 100 embodying the principles of the present invention reduces the time it takes individuals involved in the process to complete their portion of the transaction, thus reducing the overall cost of business operations. In a preferred embodiment, system 100 is a paperless procurement system which ensures that product information is always current and correct and that quick implementation of changes to catalog information, such as adding suppliers and products, is possible. A procurement system embodying the principles of the present invention provides multi-supplier item information in a single electronic catalog and allows self-service usage to the catalog user. System 100 also manages the creation of catalogs and ensures that all catalog changes are applied in an orderly manner and at the required time. It also allows catalog changes to be prepared and reviewed before they are published. New content can be staged and tested before it is published to the end-user. Once the changes are approved, those changes can be applied to the on-line production systems automatically.

Referring now to Figure 2, a block diagram of one embodiment of a site constructed according to the present invention is shown. Typically, a site, such as site 104, comprises a computer system, a single server host or a cluster of hosts and a set of clients that communicate with the server host over a network. A preferred embodiment of a site 200 comprises a processor 202, a memory 204, and a data storage device 206, an input device 208, an output device 210, and a network/internet link 212. Processor 202 is connected by a bus 214 to memory 204, data storage device 206, input device 208, output device 210, and network/internet link 212 in a Von Neuman architecture. Processor 202, memory 204, input device 208, and output device 210 may be coupled in conventional manner such as a

personal computer. Processor 202 is preferably a microprocessor such as an Intel Pentium processor; input device 208 is preferably a keyboard and/or mouse type controller; output device 210 is preferably a video monitor.

Processor 202 is also coupled to data storage device 206, such as a hard disk drive, in conventional manner. Data storage device 206 contains an information source, i.e. the databases and directories that contain information managed by an information owner. In a preferred embodiment, data storage device 206 includes a non-production database or staging system 216 and at least one production distributed relational database or on-line system 218. Staging system 216 and on-line system 218 may be located on a single hard disk drive on a single server or separate hard disk drives on separate servers connected via a network connection. In a preferred embodiment, staging system 216 and on-line system 218 are located on separate servers, such as a Pentium class server with at least 6 Gbytes of disk space, connected via a TCP/IP network connection. Staging system 216 and on-line system 218 are described below in more detail with reference to Figure 4.

Bus 214 is also coupled to network/internet link 212 to facilitate communication between site 200 and other sites of the network. In a preferred embodiment of the present invention, network/internet link 212 preferably comprises a network adapter card including a transceiver that is coupled to a cable or line. For example, network/ internet link 212 may comprise an Ethernet card connected to a coaxial cable line, a twisted pair line or a fiber optic line. Those skilled in the art will realize that a variety of different networking configurations and operating systems may be used and that the present invention is independent of such use.

Network/Internet link 212 is responsible for sending, receiving, and storing the signals sent over the network or within a protected domain of a given network. Network/Internet link 212 is coupled to bus 214 to provide these signals to processor 202 and vice versa.

Network/internet link 212 also facilitates communication between site 200 and other networks. Specifically, network/internet link 212 sends data and messages via bus 214 to other networks. For example, network/internet link 212 may include a modem, a bridge or a router coupled to the other networks in conventional manner. In a preferred embodiment of

the present invention, the network/internet link 212 is preferably a router or a high speed switch.

Processor 202, under the guidance and control of instructions received from memory 204 and from the user through input device 208, provides signals for sending and receiving data using network/internet link 212. The transfer of data between sites is broken down into packets or packages. A packet or package is a unit of data that is routed between an origin and a destination on the Internet or other network. Those skilled in the art will realize that site 200 may also be a mini-computer or a mainframe computer.

Referring now to Figure 3, a preferred embodiment of the memory 204 for site 200 is shown in more detail. Memory 204 is preferably a random access memory (RAM), but may also include read-only memory (ROM). Memory 204 includes a procurement system module 301 which preferably includes an open system interfaces module 302, a functional components module 304, a component application program interfaces (APIs) module 306, a purchase manager module 308, a procurement APIs module 310, a purchase requisition (PR) application module 312, and a PR API module 314. Memory 204 also includes an operating system 320, and an API module 322.

Open system interfaces module 302 is coupled to storage device 206 from Figure 2 and typically includes industry standard APIs to the information source, i.e. the databases and directories that contain the information managed by an information owner. In a preferred embodiment, the information source is accessed via the Internet using hypertext transfer protocol (HTTP) and is assumed to be remote from memory 204.

Functional components module 304 comprises components that provide the core functions for the invention. For example, in a preferred embodiment of the invention, functional components module 304 preferably includes functional modules such as an access control and administration module, a workflow rules module, a database access module, an information manager module, an authoring module, a release manager module, and a web driver. The operation of functional components module 304 for the present invention is described below in more detail with reference to Figure 6.

Component APIs module 306 includes APIs that abstract the functionality of the components into objects that are re-used in the construction of purchase manager module 308. Component APIs module 306 provides standardized methods for individual modules to exchange information, take actions, and apply changes to data. The methods of these APIs are specific to the module and allow other modules access to data or services provided by a module.

Purchase manager module 308 is a specialized server that enables applications for the overall procurement space. Purchase manager module 308 includes functions that are specific to procurement, including APIs to backoffice processing systems such as Oracle Financials, SAP, and BAAN. The operation of Purchase manager 308 is described below in more detail with reference to Figure 5.

Procurement APIs module 310 includes APIs to the real applications used to deliver services and functions to the end-user, such as the PR Application.

PR application module 312 is the purchase requisition application that embodies all the business rules for processing a purchase requisition, presenting all the information about goods that may be purchased, and the access control configuration for the system.

PR API module 314 includes APIs that allow custom extensions to the standard PR Application, such as method to interface to legacy systems, or to processes that access pricing information. While procurement system module 301, open system interfaces module 302, functional components module 304, component application program interfaces (APIs) module 306, purchase manager module 308, procurement APIs module 310, purchase requisition (PR) application module 312, and PR API module 314 are shown as blocks of memory 204, such modules could also be continuous portions of memory such as in a computer program.

The present invention preferably uses a conventional operating system 320 such as Windows NT. Those skilled in the art will realize how the present invention may be readily adapted for use with other operating systems.

Memory 204 may also include a variety of different APIs 322 including but not limited to computer drawing programs, word processing programs, and spreadsheet programs.

Referring now to Figure 4, a preferred embodiment of the data storage device 206, staging system 216, and on-line system 218 is shown in more detail. Staging system 216 is coupled to at least one on-line system 218. Preferably, staging system 216 is coupled to at least one on-line system 218 through a network 402, which is preferably an Extranet or intranet. Data storage device 206 may also include a backend system 404 which is preferably a legacy or other system that contains data that could be used by the procurement system module 301, or that stores output from the procurement system module 301.

Staging system 216 maintains all data entered by various creators. All data is created and modified on staging system 216 and then released to on-line system 218 at a specified time using a release process. A release process in accordance with the present invention allows a user to modify data in a staging system 216 to affect changes in an on-line system 218 for use in a collaborative data sharing environment. Thus, staging system 216 maintains data which is to be used during the release process to update an on-line system 218. The operation of such release process is described below in more detail with reference to Figures 8, 9, 10, 11, 12, and 13.

On-line system 218 maintains the most current version of the shared data in a collaborative environment. In one embodiment, one staging system 216 is used to populate multiple on-line systems 218. Each such on-line system 218 may have different contents, depending on the construction of the release process. In another embodiment, one on-line system 218 may be updated from multiple staging systems 216.

Referring now to Figure 5, a block diagram of a preferred embodiment of the purchase manager module 308 constructed according to the present invention is shown in more detail. Purchase manager module 308 includes a catalog browser module 502, a shopping basket manager module 504, a purchase requisition manager module 506, a purchasing administration module 514, a purchase order manager module 512, and a workflow rules module 510.

Catalog browser module 502 is any compliant web browser, such as Netscape Communicator, that is used to view, and search the catalog for desired items. The catalog browser module 502 also provides access to all the other functions provided by the application.

Shopping basket manager module 504 creates a shopping basket to allow a user to keep selected items until they are ready to create the purchase requisition. Shopping basket manager module 504 allows multiple shopping baskets to exist simultaneously. Shopping baskets may also be saved to become shopping lists for re-use.

Purchase requisition manager module 506 offers the ability to create, edit, delete and submit purchase requisitions to procurement system module 301. A purchase requisition as defined for the purposes of the present invention is a data structure with data that is used and processed by modules within procurement system module 301. Purchase requisition manager module 506 is also coupled to workflow rules module 510 and is used to control the sequence of actions necessary to approve a new purchase requisition. For example, purchase requisition manager module 506 may withhold approval of a purchase requisition until all required approvals are obtained. Purchase requisition manager module 506 offers an easy to use intuitive web application interface to deploy business-to-business purchasing without requiring any special knowledge relevant to webpublishing, such as database (DB) definition, servers, hypertext markup language (HTML) or common gateway interface (CGI). It offers administrative capabilities in managing the information infrastructure with respect to firewalls, security, release control and administration.

Purchasing administration module 514 allows users to define application parameters, set default values, create company-wide shopping lists and enter workflow rules module 510.

Purchase order manager module 512 accepts approved purchase requisitions and transforms them into the required purchase orders to the required destination as defined in the application parameters.

Workflow rules module 510 is the encoded form of company policy that specifies the sequence of actions required to move a new purchase requisition to a fully approved state (or purchase order) if required. Workflow rules module 510 also contains the encoded form of company policy that specifies the sequence of actions required to update staging system 216 and on-line system 218. Company policies are preferably entered by an

authorized administrator by way of workflow rules module 510 which preferably takes the form of a program executing on a stored program computer. The operation of workflow rules module 510 is described below in more detail with reference to Figure 6.

Referring to Figure 6, a detailed block diagram of a preferred embodiment of the functional components module 304 of Figure 3 constructed according to the present invention is shown. Functional components module 304 preferably includes an access control and administration module 602, a database access module 604, an information manager module 606, an authoring module 608, a workflow module 610, a release manager module 612, and a web driver 614. The routines of the present invention for performing a release process primarily include release manager module 612, database access module 604, and information manager module 606.

Access control and administration module 602 provides the means to define what actions (e.g. view, create, edit, delete) a user of the invention may execute and on which data objects those actions may be performed. Access control and administration module 602 accepts data from and transmits data to release manager module 612. Access control and administration module 602 provides enforcement of access control policies such as data in staging system 216 or on-line system 218 which a user is prohibited from editing or deleting, or users which are prohibited from viewing certain data in staging system 216 or on-line system 218. Access control module 602 thus ensures that only permitted actions for a given user are executed. Access control and administration module 602 may also be used to schedule releases for each on-line system 218 on a regular basis.

Database access module 604 is the functionality that allows any other modules to create, edit, view or delete information from a database, such as staging system 216 or on-line system 218, that the present invention uses to store data. Database access module 604 accepts input from release manager module 612 and provides access to the desired database, such as staging system 218 or on-line system 216. Database access module 604 then performs the required actions (e.g. create, edit, view, delete) requested by release manager module 612 to data in the accessed database.

Information manager module 606 allows users to define, create, edit and delete objects and values associated with all data items or objects that are required for the present

invention to execute correctly. Release manager 612 accesses information manager module 606 using defined APIs to determine which data items or objects have changes pending and to retrieve the information for creating, modifying or deleting the data item. On an online system 218, release manager 612 calls information manager module 606 to apply changes to the data items.

Authoring module 608 allows a user to define web page templates for viewing the catalog or other data. Authoring module 608 stores data for a page, such as a web page, in the system using information manager module 606 and database access module 604. Release manager 612 acts on data produced by authoring module 608 using information module 606 and database access module 604 as described above.

Workflow module 610 is coupled to workflow rules module 510 and release manager module 612 and executes the business process and company policies defined by the rules of workflow rules module 510. Workflow module 610 can be setup via business rules to determine the sequence of events and timing of events in a release. The workflow module 610, using rules 510, determine whether required approvals have been given for a specific release. If not, the release will not be applied. If approved, the release is applied per the scheduling. If no rules are defined, the release process will proceed as scheduled. Thus, the present invention advantageously allows the release manager module 612 and the release process to operate according to implemented business policies.

Release manager module 612 is a component for controlling transfers of data to an on-line system 218 from a staging system 216 via a network/internet link 402 in order to create, modify, or delete data in the on-line system 218. Release manager module 612 embodies the logic to scan data in a staging system 216 that is scheduled to be released, creates a release package, moves the release package to a target online system 218 and applies changes in the release package to the target on-line system 218. Release manager module 612 uses database access module 604 and information manager module 606 to collect all changes or updates in a staging system 216 for a given release process and builds a release package using the collected changes or updates. Release manager module 612 then uses database access module 604 and information manager module 606 to apply the changes to an on-line system 218, in other words. to create, modify or delete data items in

on-line system 218 based on the changes in the release package. The operation of release manager module 612 and the release process is described below in more detail with reference to Figures 8, 9, 10, 11, 12, 13 and 14.

Web driver 614 is the interface between purchase manager module 308 and the Web environment. Web driver 614 accepts Web originated requests and HTTP posts and converts those requests into the form needed by the internal servers and processes. Web driver 614 also performs the reverse process of converting application responses into Web formats that can be displayed by the Web Browsers. Web driver 614 allows modules such as authoring module 608, access control and administration module 602, and information manager module 606 to access and take actions on data items in staging system 216.

Figure 7 of the drawings is a block diagram graphically showing the software architecture for the procurement system of the present invention. The software architecture includes a visual manager module 702, access control and administration module 602, a session manager module 706, a transaction manager module 708, procurement APIs module 310, purchase manager module 308, operating system 320, a database server 618, a Web server 720, Web driver 614, and release manager module 612. Visual manager 702 manages the information system and provides a familiar and easy to use interface to the user. It is used for creating, managing and authoring information, setting security and other features. Creation of the data definition is managed through Visual manager 702 using a familiar desktop metaphor, thus hiding the complexity of the database interface. Data loading can also be managed remotely using Visual manager 702. Visual manager 702 seamlessly handles management of all data elements. For example, the image associated with a particular category is modified on the client desktop. This image can be loaded into the database from Visual manager 702 using the web browser. Bulk loading of data as well as interfacing to other database systems is also provided. Users can also extend the system and customize it to their needs by adding properties to categories and items through Visual manager 702. Properties can be global or local and shared by different logical entities. Session manager 706 provides a logical context for a connection to the application and maintains that context for the duration of the session. Transaction manager 708 ensures that any operation that needs to be executed only once and must be complete as an atomic operation is executed correctly and completely, or is un-done if it fails for any reason.

Operating system 320 is any conventional operating system, such as the Windows NT operating system. Database server 718 is a standard database server such as an Oracle workgroup server. Web server 720 is a standard web server such as the Netscape Enterprise server.

A preferred method of operation and an embodiment for release manager module 612 will be described focusing on its relationship to and its control of staging system 216 and target on-line system 218. Referring to Figure 8, a flow chart of a preferred method for performing a release process in accordance with the present invention is shown. The release process provides for the modification of data, also referred to as data items, records or objects, in staging system 216 to affect changes in on-line system 218 for use in a collaborative data sharing environment. The procedure is entered at 800. The staging system 216 is updated (802) by a user of the system by creating, modifying, editing or deleting each record or data in the staging system 216 that will be updated in the on-line system 218 during a release process. The method for updating staging system 216 is described below in more detail with reference to Figure 9. After each record or data item has been updated, the updated record or data, ("update") is transferred to the packaging area. The packaging area is generally a file in memory used to store the update, and in one embodiment, the packaging area is a temporary file in staging system 216 which is used exclusively by release manager module 612. Each update includes all the information needed to identify where that updated record or data is to be applied and how and when it is to be applied in the target on-line system 218. In a preferred embodiment, each update contains a plurality of changed fields, such as a source item identifier for the source of the update, a destination item identifier for the destination of the update, an action field specifying the action to be taken, a body field containing textual and/or graphics data embodying the update, and a release date indicating when the action to be taken is to take place. The update is then packaged (804) by release manager module 612 into a package of files. Each file in the package contains updates of the same type. For example, in a preferred embodiment, each file in the package contains updates which are grouped together based on item type and action to be performed. A package is built once and applied to each target on-line system 218 by release manager module 612. Once a package is built, it can be applied to multiple target online systems connected to the same staging system. The method for packaging the update is described below in more detail with reference to Figure 11. The packaged update

is then transferred (806) by release manager module 612, preferably using File Transfer Protocol (FTP), from the staging system 216 to the target on-line system 218. The method for transferring packaged data to target on-line system 218 is described below in more detail with reference to Figure 12. There may be only one target on-line system or there may be several target on-line systems receiving the packaged update. After the packaged update has been successfully transferred to the target on-line system 218, the target on-line system 218 is updated (808) by release manager module 612. The method for updating on-line system 218 is described below in more detail with reference to Figure 13.

A release of data from a staging system 216 to an on-line system 218 can be of two fundamental types: a full release, or an incremental release. A full release transfers a copy of the current data in the staging system 216 in its entirety to the online system 218. A full release occurs when the entire data of an on-line system 218 is replaced by the new data from a staging system 216. Preferably, this type of release must occur at least once before an incremental release can be made. For example, when a new on-line system 218 is initialized, the first release is a full release. An incremental release transfers a sub-set of data in the staging system 216 to the on-line system 218. An incremental release occurs at a scheduled time and makes specific changes to one or more on-line systems 218. The changes can be additions, deletions or updates to existing data. Incremental releases can also be run ad hoc if required to address urgent changes outside the pre-set schedule. Release manager module 612 generally controls all time related functions, and in one embodiment, a standard operating system service, such as KRON, is used to schedule releases at the defined time intervals. The release process of the present invention is described in more detail with reference to Figures 9, 10, 11, 12 and 13.

Release manager module 612 performing the release process advantageously allows many different changes to be made concurrently to data shared by different creators of data. After the initial creation and installation of data on on-line system 218, the release process advantageously provides for the addition, deletion or modification of the data on on-line system 218 in an ordered manner to ensure that the data remains current, correct, and consistent. Moreover, the release process of the present invention advantageously provides for a given change to data on on-line system 218 to be implemented on a given date, also known as a "release date," and also provides that one or many changes may be scheduled to

be implemented on the release date. Thus, the release process advantageously manages the multiple concurrent changes to shared data on on-line system 218, ensures the consistency of such data, and prevents the corruption of such data which may result from conflicting changes made by different creators to the same data.

In a preferred embodiment, procurement system module 301 includes release manager module 612 to perform a release process in the automation of the procurement and purchase of items. In such embodiment, on-line system 218 is the module through which users retrieve information, place orders, and otherwise process orders. The present invention operates such that there may be a plurality of such on-line systems 218. In such embodiment, staging system 216 is the module to which the authors of data in the catalog, or the authors of processes for handling orders have access. These authors can modify the data in staging system 216 and specify when such changes should be made to any one of a number of on-line systems 218 that are in existence. Procurement system module 301 which includes release manager module 612 controls the updating of data on the on-line systems 218 and can perform incremental updating of data to only portions of the on-line system 218 or full updating of data where all modified portions of the on-line system 218 are made current. A procurement system module 301 including a release process advantageously provides for the modification of shared data in a collaborative environment such as the creation of and modification to a catalog of items that may be accessed and purchased by various users.

Figure 9 of the drawings is a flowchart of a preferred method for updating the staging system 802 as shown in Figure 8. The procedure is entered at 900. The staging system 216 is scanned (902) for records or data that will be changed for a given release process. As each record or data is located, the update is applied (904) to the staging system 216. At this stage, the status of the process is reported (906) to a staging log file by release manager module 612. The staging log file generated by release manager module 612 can be viewed by access control and administration module 602. The staging log file indicates the status of a particular stage of the release process and in this procedure indicates that the staging update has started and that the update is in progress. Typically, the staging log file is a standard file system kept in a system designated directory. The procedure then checks to determine whether the update completed correctly (908) by checking whether the scheduled

change was successfully applied to the staging system 216 data. If the update was successfully completed, then the status reported indicates that the staging update is completed (916). The successfully completed update is then reported (918) in the staging log file by release manager module 612, and the record indicating the matching change for the on-line system 218 is transferred (920) to the packaging area. If an error occurred during execution and the update did not complete correctly, the status reported indicates that the staging update has been aborted (910). The reason for the failure is recorded (912) in the staging log file. The update is then reexecuted (914) and the update is re-applied (904) to the staging system 216. Once a record or data has been successfully updated, the procedure determines whether another record or data needs to be updated (920). Release manager module 612 continually scans the staging system 216 to check for items that have scheduled changes for a specific date and time. Once all changes for a given release have been found, the process is complete. If another record or data item is to be updated for the given release process, steps 902 through 920 are repeated. If all records that need to be updated nave been successfully updated, the process packages the update as described below in more detail with reference to Figure 11. Thus, the present invention advantageously time stamps the changes which are to be applied to a target on-line system 218 indicating when the changes should be applied and hence determines a specific sequence for changes in an on-line system 218. Additionally, changes can be made both to items inside the database as well as to those parts of the item not located in the database, i.e. non-database parts. These non-database parts are usually large binary objects such as images, documents, video or sound.

Referring to Figure 10, a flowchart of a preferred method for maintaining data consistency during the release process in accordance with the present invention is shown. The method ensures that regardless of data type, only one change is scheduled for a particular data item or record on the release date, and that once a record or data is marked as "to-be-deleted," no further changes are scheduled after the "to-be-deleted" date. The procedure is entered at 1000. The record or data is accessed (1002) and locked (1004) using database access module 604 to prevent others from accessing or editing the record or data. The procedure then determines whether the record or data is already marked for deletion (1006) by checking the action required and the required date of the update. If the record or data is marked for deletion, then no modifications may be made to the record or data and the procedure ends. If the record or data is not marked for deletion, the record or data is

accessed to determine the modifications and release dates (1008). The release date for this modification is then determined (1010). The procedure then determines whether the record or data is marked for change on the release date for this modification (1012). If the record or data is marked for change on the release date for this modification, then no further changes are scheduled for the particular record or data and the procedure ends. If the record or data is not marked for change on the release date for this modification, then the modification is stored (1014) to the record or data in the staging system, and the record or data status fields are updated (1016) by release manager module 612 through database access module 604.

Figure 11 of the drawings is a preferred method for packaging the update 804 as shown in Figure 8. The procedure is entered at 1100. The release date for the given release process is determined (1102) from the data. Releases may be scheduled to be an automated release, such as a daily release, or to be a manually run release, in which case an operator runs the release . When a release process for a specified release date is run, all scheduled changes since the previous completed release process and the current date will be applied in sequence. All updates in the staging system 216 scheduled for release on the specified date are then identified (1104) and extracted (1106) from staging system 216 by release manager module 612 which copies all updates with scheduled changes to a temporary package file. The extracted updates are then packaged (1108) into a file. During this stage, release manager module 612 reports the status to a staging log file to indicate that the packaging has started (1110). The procedure then determines whether the packaging completed correctly (1112) by verifying that all updates with current pending actions are recorded in the package. If the packaging was successfully completed, the status reported indicates that the packaging has completed (1120). The successfully completed package is then recorded (1122) in the staging log file. If an error occurred and the update was not completed correctly, the status reported indicates that the packaging has been aborted (1114). The reason for the failure is recorded (1116) in the staging log file. The packaging run is then reexecuted by repeating steps 1108 through 1112. After successful completion of the packaging, the procedure then determines whether any more updates need to be packaged (1124) by scanning staging system 216 for updates with pending actions on or before the specified date for the given release process. If another update is to be packaged for the given release process, steps 1102 through 1122 are repeated. If all updates that need to be packaged have been successfully packaged, the process commences with the transfer phase.

The present invention advantageously allows updates to both database data and non-database data in the update procedure.

Figure 12 is a flowchart of a preferred method for transferring packaged updates to a target on-line system 806 as shown in Figure 8 in accordance with the present invention. The procedure is entered at 1200. The targeted on-line system for the scheduled release is located (1202) and the packaged update or package is transferred to the targeted on-line system. As discussed above, there may be more than one targeted on-line system. In such cases, each targeted on-line system would be located by release manager module 612 and access control and administration module 602. Access control and administration module 602 is usually used to regularly schedule releases for each on-line system and release manager module 612 will be invoked at each scheduling to perform the release process. The package is then transferred to each targeted on-line system. Preferably, the transfer is completed using FTP. Typically, release manager module 612 invokes an FTP process to execute the file transfer. At this stage, the status of the process is reported (1206) indicating that the transfer has started and is in progress. Release manager module 612 then checks to determine whether transfer of the package was successfully completed (1208). If the transfer was successfully completed, then the status is reported indicating that the transfer has been completed (1216) and the successfully transferred package is recorded in the staging log file (1218). If an error occurred during execution and the package did not transfer correctly, the status reported indicates that the transfer has been aborted (1210). The reason for the failure is recorded (1212) in the staging log file. The transfer is then re-executed (1214) and the package is re-sent to the target on-line system (1204). Once a package has been successfully transferred, the procedure determines whether another package needs to be transferred (1220). If another package is to be transferred for the given release process, steps 1202 through 1218 are repeated. If all packages for the given release process have been successfully transferred, the release process commences with the online update.

Figure 13 is a flowchart of a preferred method for an on-line updating process 808 of Figure 8 in accordance with the present invention. After the package has been successfully transferred to the target on-line system, the on-line update phase can begin. The procedure is entered at 1300. The packaged update is opened (1302) by release manager module 612 and the record or data item in the on-line system which is to be updated is selected (1304). The

selected record or data item to be updated is then locked (1306) by release manager module 612 using database access module 604 to prevent others from accessing the record during the on-line update. The record remains locked throughout the duration of the update. The update is then applied (1308) to the on-line system using the packaged update. In other words, the changes which were scheduled for the given release date are made to the data in on-line system 218. The status reported by release manager module 612 indicates that the online update has started and is in progress (1310). The procedure then determines whether the update was completed successfully (1312). If the update was completed successfully, then the status reported indicates that the staging update has been completed (1320). The successfully completed update is then recorded (1322) in the staging log file. If an error occurred during execution and the update did not complete correctly, then the status reported indicates that the on-line update has been aborted (1314). The reason for the failure is recorded (1316) in the staging log file. The on-line update is then re-executed (1318) and the update is re-applied to the on-line system (1308). Once a record in the on-line system has been updated successfully, the procedure determines whether another record or data needs to be updated (1324). If another record or data is to be updated for the given release process, steps 1304 through 1322 are repeated. If all records in the on-line system that need to be updated have been updated successfully, the update package is closed (1326).

While the present invention has been described with reference to certain preferred embodiments, those skilled in the art will recognize that various modifications may be provided. These and other variations upon, and modifications to the preferred embodiment are provided for by the present invention, which is limited only by the following claims.